



جامعة طرابلس
كلية تقنية المعلومات

خريف 2019

البرمجة الشيئية

Object Oriented Programming
(with Java)

رمز المقرر: ITGS211

محاضرة التجريد: Abstraction

الفصل الدراسي: خريف 2019

التجريد Abstract

التجريد : هو عملية تقوم بتعريف السلوك والمهام التي يقوم بها الكائن (object) وإهمال التفاصيل غير اللازمة.

مثال :



جهاز الراديو لديه هوائي وزر تحكم بالصوت وزر للفتح والإغلاق، ولا نحتاج لمعرفة كيف يلتقط جهاز الراديو الموجات من الإذاعة وكيف يقوم بتحويلها إلى إشارات رقمية ثم يخرجها في شكل صوت، فهذه التفاصيل مخفيه عنا ولا تحتاجها. إخفاء التفاصيل الداخلية تعتبر ميزة فبهذه الطريقة يمكن لكل شخص أن يستخدم جهاز الراديو ولا يقتصر الأمر على الفنيين فقط.

2



التعقيد الذي داخل جسم الفيل أو اي حيوان آخر لا يتم التعرض له إطلاقاً ولكن يتم التعامل معه كحيوان له حركات ووظائف وصفات ظاهرة فقط.

أيضاً نعلم أن السيارة تسير و لكن لا تتعرض لكيفية انتاج الحركة. فعند النظر للسيارة ننتبه فقط لكونها سيارة و إذا دققنا النظر ننتبه للونها وعدد أبوابها وشكلها، ولكن لا يلفت انتباهنا كيفية سيرها وطريقة وصول الوقود للمحركات وطريقة توقفها. فهذا التعقيد كله لا نتعامل معه في حياتنا.







3

مثلاً: عند عمل برنامج لشركة ما، وكانت الكائنات التي نتعامل معها ثلاث:-

- مدير Manager و موظف Employee و عميل Client

Manager	Employee	Client
Emp_Id	Emp_Id	Client_Id
Name	Name	Name
Age	Age	Age
Address	Address	Address
Degree	Degree	
Salary	Salary	
Commission		
Print_Info()	Print_Info()	Print_Info()

لاحظ أن هناك خصائص مشتركة بين المدير والموظف والعميل وهي كونهم كلهم شخص له اسم **Name** وعمر **Age** وعنوان **Address** من هنا يمكن انشاء صنف شخص **person class** يحوي الخصائص المشتركة وترثه الأصناف الثلاث كالتالي:

person
Name
Age
Address
Print_Info()

4

person

Name

Age

Address

Print_Info()

Manager

Emp_Id

Degree

Salary

Commission

Employee

Emp_Id

Degree

Salary

Client

Client_Id

لاحظ

لا زالت هناك خصائص مشتركة بين المدير والموظف فكلهما موظف ولكن يتميز المدير عن الموظف العادي بالعمولة Commission.

5

person

Name

Age

Address

Print_Info()

Employee

Emp_Id

Degree

Salary

Print_Info()

Client

Client_Id

Print_Info()

Manager

Commission

Print_Info()

person

Name

Age

Address

Print_Info()

Manager

Emp_Id

Degree

Salary

Commission

Employee

Emp_Id

Degree

Salary

Client

Client_Id

بما أن هناك خصائص مشتركة بين المدير والموظف سيتم انشاء صنف مدير يرث صنف الموظف Employee كما التالي:-

6

Abstract Class

➤ في classes السابقة كان بإمكاننا اشتقاق كائنات منها والتعامل معها، هذه

classes تسمى Concrete class.

➤ هناك نوع آخر من classes لا يُسمح بأشتقاق كائنات منه بل يتم فقط اتخاذه كأب (Superclass) وهذه تسمى classes المجردة Abstract class.

➤ يتم تعريف class من النوع المجرد بإضافة كلمة Abstract قبل الكلمة المحجوزة class .

```
public abstract class Employee {  
    //....  
}
```

➤ يتم تعريف class من النوع المجرد بإضافة كلمة Abstract قبل الكلمة الغرض منه أن يتم توفير class عام يمكن للأبناء subclasses أن يشتقوا منه، ويشتركوا جميعاً في تصميم واحد، لكن لكل واحد منهم طريقة ما للتطبيق والعمل.

7

Abstract Class

➤ class المجرد يحتوي على دالة مجردة أو أكثر. وهذه الدوال يجب أن يتم عمل override لها في الابناء Subclasses لكي تصبح الابناء concrete classes.

➤ الدوال والمتغيرات غير المجردة في class المجرد تخضع لقواعد الوراثة العامة عند توريث هذا الصنف للأبناء.

➤ محاولة اشتقاق كائن من class المجرد ينتج عنها Compilation error.

```
10 | x1 is abstract; cannot be instantiated  
11 | ----  
12 | (Alt-Enter shows hints)  
13 |  
14 |     x1 xx = new x1();  
15 |     }  
    |  
    | abstract class x1{
```

➤ مثال لاستخدام الصنف المجرد Abstract Class:

يمكننا كتابة Abstract Class لتمثيل الأشكال الهندسية ثنائية الأبعاد، ثم نشق منه concrete classes للمربع والدائرة والمستطيل.

8

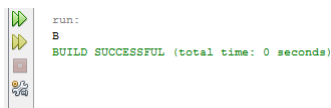
Abstract Class

من المفترض أن الصنف المجرد Abstract class يحوي على الأقل دالة من النوع المجرد Abstract method.
ملاحظة: إن لم يحوي دالة مجردة فلن يظهر خطأ ولكن ما الفائدة منه؟!

```
public class Abs_JavaApp {
    public static void main(String[] args) {
        B b=new B();
        b.printA();
    }

    abstract class A{
        int M;
    }

    class B extends A{
        void printA() {
            System.out.println("B");
        }
    }
}
```

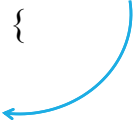


الدالة المجردة Abstract method

من المفترض أن الصنف المجرد Abstract class يحوي على الأقل دالة من النوع المجرد Abstract method.
والدالة المجردة هي دالة يتم تعريفها فقط في class دون عمل تطبيق لها، حيث يتم عمل التطبيق من خلال الابناء subclass مستخدمين مفهوم overridden.

أي يوضع method signature فقط متبوعاً بفاصلة منقوطة (;) أي بدون method body كالتالي:

```
public abstract class Employee {
    //---
    Public abstract void cal();
}
```



الدالة المجردة Abstract method

لو حاولت تكتب آلية عمل الدالة المجردة فسيظهر لك خطأ كالتالي:

```
abstract methods cannot have a body
----
(Alt-Enter shows hints)
abstract void printA() {
    System.out.println("A");
}
```

لو كتبت الدالة المجردة بشكل صحيح ولكن لم نعمل لها overriding فسيظهر لك خطأ كالتالي:

```
abstract class A{
    int M;
    abstract void printA();
}

class B extends A{
    void printB() {
        System.out.println("B");
    }
}
```

B is not abstract and does not override abstract method printA() in A

(Alt-Enter shows hints)

الدالة المجردة Abstract method

➤ إذا كان class المجرد abstract class فهذا يعني أنه لابد من وراثة هذا class. والدالة المجردة Abstract method لابد من عمل دالة بنفس اسمها في Sub class. أي overriding كما في المثال التالي:

```
abstract class x1{
    abstract void show();
}

final class x2 extends x1 {
    @Override
    void show() {
        System.out.print("Hi");
    }
}
```

إذا وضعت الأمر abstraction أمام دالة في class لابد من وضع الأمر abstraction أمام class الذي يحوي هذه الدالة وإلا سيظهر خطأ كالتالي:

```
x1 is not abstract and does not override abstract method show() in x1
----
(Alt-Enter shows hints)

class x1{
    abstract void show();
}
```

```
public class Abstract1 {
    public static void main(String[] args) {
        second s = new second();
        s.show();
        s.showtext();
    }

    abstract class first{
        abstract void show();
        void showtext()
        {
            System.out.println("first");
        }
    }

    class second extends first{
        @Override
        void show(){
            System.out.println("second");
        }
    }
}
```

second

first

13

```
public class Abstract1 {
    public static void main(String[] args) {
        second s = new second();
        first f = new first();
        s.show();
        s.showtext();
    }

    abstract class first{
        abstract void show();
        void showtext()
        {
            System.out.println("first");
        }
    }

    class second extends first{
        @Override
        void show(){
            System.out.println("second");
        }
    }
}
```

ماهو الخطأ عند تعريف كائن من class مجرد (Abstract) ؟؟

4
6
7
8

second s = new second();
first f = new first();
first is abstract; cannot be instantiated

(Alt-Enter shows hints)

14

إذا كان superclass مجرد ويحوي دالة مجردة Abstract ،، ولكن لم يتم إنشاء دالة بنفس الاسم في subclass فستظهر رسالة خطأ كالتالي:

```
3 public static void main(String[] args) {
4     second s = new second();
5     s.show();
6     s.showtext();
7 }
8
9 abstract class first{
10     abstract void show();
11     void showtext()
12 {
13
14 second is not abstract and does not override abstract method show() in first
15 ----
16 (Alt-Enter shows hints)
17
18 class second extends first{
19     void showAB() {
20         System.out.println("second");
21     }
22 }
```

15

Example :

قم بإنشاء class باسم x1 مع منع إستعماله مباشرة، علماً بأن x1 يحتوي على دالة مجردة max تعمل على تحديد أكبر رقم من بين ثلاث أرقام. و x1 هو اب لـ x2 الذي يحوي دالة numbers التي تعمل على استقبال ثلاث أرقام، ولا ترجع أي قيمة. وعليك منع توريث x2 لأي classes أخرى. في x2 قم بالتركيب على الدالة المجردة (max) بحيث تعمل على تحديد أكبر رقم من ثلاث أرقام .

أكتب الجمل البرمجية لإستعمال الـ class x2 من أجل طباعة قيمة العدد الأكبر وذلك من خلال استدعاء الدالة numbers , max.

Superclass

x1 class

Subclass

X2 class

16

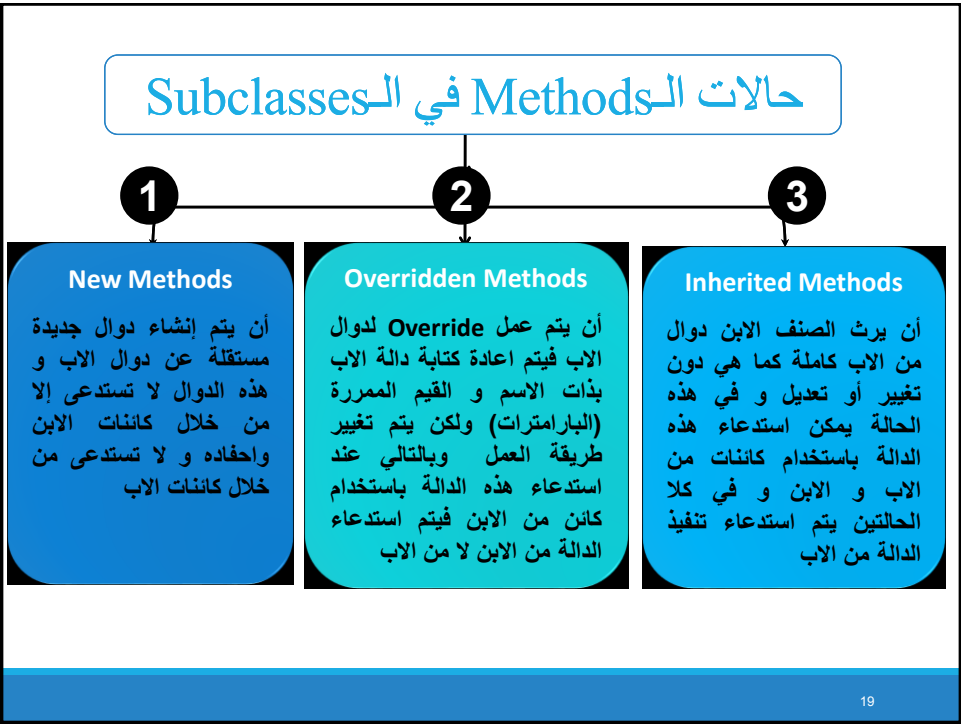

```
import java.util.Scanner;
public class Myclass {
    public static void main(String[] args) {
        x2 z= new x2();
        z.numbers();
        System.out.println("max is : " + z.max(z.n1,z.n2,z.n3));
    }
}
```

```
abstract class x1
{
    abstract int max (int a, int b , int c);
}
```

17

```
final class x2 extends x1 {
    int n1, n2, n3;
    void numbers(){
        Scanner in = new Scanner (System .in);
        n1= in.nextInt();
        n2= in.nextInt();
        n3= in.nextInt();
    }
    @Override
    int max (int a, int b, int c) {
        int result ;
        if (a>b && a>c)
            result = a;
        if(b>a && b>c)
            result =b;
        else
            result =c;
        return result ;
    }
}
```

18



إنشاء Abstract Superclass Employee

```
public abstract class Employee{  
    private String firstName;  
    private String lastName;  
    private String ID;  
  
    public Employee( String firstName, String lastName, String ID )  
    {  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.ID = ID;  
    } // end three-argument Employee constructor  
  
    // set firstName name  
    public void setfirstName( String firstName )  
    {  
        this.firstName = firstName;  
    } // end method setfirstName  
  
    // return firstName name  
    public String getfirstName()  
    {  
        return firstName;  
    } // end method getfirstName  
  
    // set last name  
    public void setLastName( String lastName )  
    {  
        this.lastName = lastName;  
    } // end method setLastName  
  
    // return last name  
    public String getLastName()  
    {  
        return lastName;  
    } // end method getLastName  
}
```

انتبه قمنا بتعريف هذا الصنف class بالكلمة **abstract** وبالتالي من المفترض أن يحتوي على دالة واحدة على الأقل من النوع **abstract**

20

إنشاء Abstract Superclass Employee

```
// set last name
public void setLastName( String lastName )
{
    this.lastName = lastName;
} // end method setLastName

// return last name
public String getLastName()
{
    return lastName;
} // end method getLastName

// set ID
public void setID( String ID )
{
    this.ID = ID;
} // end method setID

// return ID
public String getID()
{
    return ID;
} // end method getID

// return information of Employee object
public String info()
{
    return "The information is: " + getfirstName() + getLastName() + getID();
} // end method info

// abstract method overridden by subclasses
public abstract double earnings(); // no implementation here
} // end abstract class Employee
```

انتبه: هذه الدالة المجردة (أي مجردة من التطبيق) ويتم تعريفها فقط من خلال التوقيع method signature (نوعها، اسمها، قيمها). وتذكر أن من دونها فإن هذا الصنف لا يعتبر مجرد

21

إنشاء Concrete Subclass SalariedEmployee

```
public class SalariedEmployee extends Employee
{
    private double weeklySalary;

    // four-argument constructor
    public SalariedEmployee( String firstName, String lastName, String ID,
                           double weeklySalary )
    {
        super( firstName, lastName, ID ); // pass to Employee constructor
        this.weeklySalary = weeklySalary ;
    } // end four-argument SalariedEmployee constructor

    // set salary
    public void setWeeklySalary( double weeklySalary )
    {
        this.weeklySalary = weeklySalary < 0.0 ? 0.0 : weeklySalary;
    } // end method setWeeklySalary

    // return salary
    public double getWeeklySalary()
    {
        return weeklySalary;
    } // end method getWeeklySalary
}
```

استدعاء صريح للـ constructor الخاص بالأب بقيم ممررة عبر الكائن المشتق من الابن.

22

إنشاء Concrete Subclass **SalariedEmployee**

هنا قمنا بعمل override للدالة earnings من خلال:
تغيير طريقة حساب المرتب بينما احتفظنا بتوقيع الدالة كما هو

```
// calculate earnings; override abstract method earnings in Employee
public double earnings()
{
    return getWeeklySalary();
} // end method earnings

// return String representation of SalariedEmployee object
public String info()
{
    return super.info() + getfirstName() + getLastName() + getID() ;
} // end method info

} // end class SalariedEmployee
```

هنا قمنا بعمل override للدالة info من خلال:
تغيير طريقة إرجاع البيانات واستدعاء دالة الاب أيضا من خلال super.info()

23